

On the computational complexity of minimum-concave-cost flow in a two-dimensional grid

Shabbir Ahmed*, Qie He†, Shi Li‡, George L. Nemhauser§

Abstract

We study the minimum-concave-cost flow problem on a two-dimensional grid. We characterize the computational complexity of this problem based on the number of rows and columns of the grid, the number of different capacities over all arcs, and the location of sources and sinks. The concave cost over each arc is assumed to be evaluated through an oracle machine, i.e., the oracle machine returns the cost over an arc in a single computational step, given the flow value and the arc index. We propose an algorithm whose running time is polynomial in the number of columns of the grid, for the following cases: (1) the grid has a constant number of rows, a constant number of different capacities over all arcs, and sources and sinks in at most two rows; (2) the grid has two rows and a constant number of different capacities over all arcs connecting rows; (3) the grid has a constant number of rows and all sources in one row, with infinite capacity over each arc. These are presumably the most general polynomially solvable cases, since we show the problem becomes NP-hard when any condition in these cases is removed. Our results apply to abundant variants and generalizations of the dynamic lot sizing model, and answer several questions raised in serial supply chain optimization.

1 Introduction

We are interested in the min-cost network flow problem over an L -by- T grid G . The grid G is an acyclic directed planar graph on vertices $V = \{v_{l,t} \mid l = 1, \dots, L; t = 1, \dots, T\}$ with arc set

$$A = \{(v_{l,t}, v_{l+1,t}) \mid l \leq L-1\} \cup \{(v_{l,t}, v_{l,t+1}) \mid t \leq T-1\}.$$

A drawing of G , embedded in \mathbb{R}^2 , is shown in Figure 1. The grid G has some additional parameters: a supply vector $b = (b_v) \in \mathbb{Z}^{|V|}$, a cost function $c_a : \mathbb{R} \rightarrow \mathbb{R}$ for each arc $a \in A$, and an arc-capacity vector $U = (U_a) \in \bar{\mathbb{Z}}_+^{|A|}$, where $\bar{\mathbb{Z}}_+$ is the set of non-negative integers and infinity. We assume the net supply $\sum_{i \in V} b_v = 0$. We call a vertex v a source if $b_v > 0$, a sink if $b_v < 0$, or a transshipment vertex otherwise. We assume the function c_a over each arc a is represented by an oracle machine. Given the flow value x over arc a , the oracle machine returns the cost $c_a(x)$ in a single computational step, so cost functions with explicit forms are special cases in our oracle model.

*H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta GA 30332, USA. Email: sahed@isye.gatech.edu.

†Department of Industrial & Systems Engineering, University of Minnesota, Minneapolis MN 55455, USA. Email: qhe@umn.edu.

‡Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY 14260, USA. Email: shil@buffalo.edu.

§H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta GA 30332, USA. Email: gnemhaus@isye.gatech.edu.

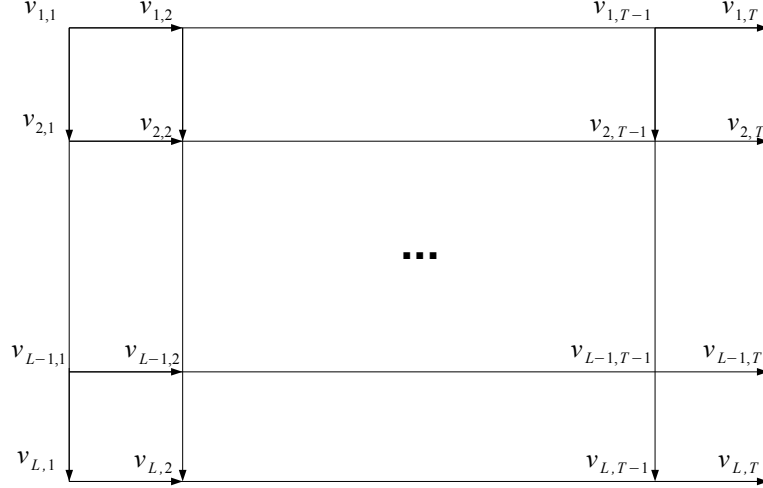


Figure 1: The L -by- T grid

The minimum-concave-cost flow problem in a two-dimensional grid (MFG) is to find a vector $x \in \mathbb{R}^{|A|}$ to

$$\begin{aligned} & \text{minimize} && \sum_{a \in A} c_a(x_a) \\ & \text{s.t.} && \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = b_v, \quad \forall v \in V, \\ & && 0 \leq x_a \leq U_a, \quad \forall a \in A, \end{aligned} \tag{1}$$

where c_a is concave for each $a \in A$, and $\delta^+(v)$ and $\delta^-(v)$ are the set of outgoing and incoming arcs of vertex v , respectively. A *feasible flow* is a vector $x \in \mathbb{R}^{|A|}$ that satisfies the constraints in (1).

Our interest in the computational complexity of the MFG is kindled by the classical dynamic uncapacitated lot sizing problem (ULSP), a building block in production planning and inventory control, studied by Wagner and Whitin [29]. The ULSP aims to find a min-cost production schedule to satisfy a given sequence of demands over T periods, with no limit on production capacity. It is a special case of the MFG with two rows and one source [17], and can be solved by dynamic programming (DP) efficiently in $O(T \ln T)$ time [1, 11, 28]. Many efforts have been spent on extending the ULSP to models under more practical settings, such as multiple production steps, demands at intermediate production steps, capacity on production or storage, and different production and storage costs which characterize the setup cost or economies of scale. All these variants can be transformed into the MFG with *a single source* by adjusting the corresponding parameters in (1), such as L , U_a , and the location of sinks [17]. The computational complexity, however, varies across models. For example, the constant capacitated lot sizing problem, an extension of the ULSP with a uniform production capacity at all periods, can be solved in $O(T^3)$ time [26]; In contrast, the capacitated lot sizing problem, where the production capacities can be arbitrary across periods, is NP-hard. Let

$$K = |\{U_a \mid U_a < +\infty, a \in A\}|$$

be the number of different arc capacity values of the grid. Our goal is to study how the network parameters, in particular *the number of rows L , the number of arc capacity values K , and the location of sources and sinks*, affect the computational complexity of the MFG. In particular, we aim to identify the most general conditions under which the MFG can be efficiently solved.

Since the arc capacity plays such a significant role in the computational complexity of the MFG, we divide the MFG into two classes: the uncapacitated MFG (U-MFG), where each arc has infinite capacity or $K = 0$, and the capacitated MFG (C-MFG), where *at least* one arc has a finite capacity

or $K \geq 1$. We summarize the complexity of the two classes in Tables 1 and 2 below, with newly derived results in this paper highlighted in bold.

Table 1: Complexity of the C-MFG with constant K

	Sources and sinks in at most two rows	Sources and sinks in at least three rows
Constant L	Polynomially solvable	NP-hard
Varying L	Open	NP-hard

Table 2: Complexity of the U-MFG

Sources in one row			Both sources and sinks in at least two rows
Constant L	Sinks in one row	Polynomially solvable [10]	NP-hard
	Sinks in at most two rows	Polynomially solvable [17]	
	Sinks in at most L rows	Polynomially solvable	
Varying L	Sinks in at most L rows	NP-hard	NP-hard

In Table 1, we only list the results for constant K , since with varying K the capacitated lot sizing problem, a special case of the C-MFG, is already NP-hard. As we see from the two tables, each condition for the polynomially solvable cases is crucial; any relaxation of these conditions renders the problem NP-hard. Our result answers several questions raised in the context of supply chain optimization. In [19, 27], polynomial-time algorithms were derived for a multi-stage serial supply chain design problem, in which capacity is only present at the manufacturing stage (the vertical arcs between the first and second rows in the MFG); the authors raised the question of whether an optimal design can be found efficiently, if capacity is imposed at other stages. Our result in Table 1 indicates that the optimal design can always be found in polynomial time, as long as the number of different capacities is a constant, regardless of at which stages the capacity is imposed. Another open problem was raised in [17] on whether the U-MFG with sources in one row and sinks in $L > 2$ rows is polynomially solvable. In this paper, we give a positive answer to the problem when L is an arbitrary positive integer, thus generalizing the result in [17] from sinks in at most two rows to sinks in an arbitrary number of rows; we also complement the polynomially solvable case with a hardness result when L is allowed to vary.

We should also mention the model of computation used in this paper. Since the cost over an arc in a feasible flow may be a real number, any algorithm for the MFG will involve computation over real numbers. We adopt the real Random Access Machine (RAM) introduced in [8] as our model of computation; the real RAM model has registers each of which is capable of storing a real number with infinite precision, and each arithmetic operation on real numbers stored in these registers, such as adding or comparing two real numbers, takes a single computational step. The running time of our algorithm refers to the total number of arithmetic operations and oracle queries.

We now survey some results on minimum-concave-cost flow problem in the literature. The minimum-concave-cost flow problem over a general network can be shown to be NP-hard, proven by a reduction from the partition problem [15]. Approximation and exact algorithms based on DP or branch-and-bound were developed for various network topologies and cost functions; see the survey in [15] and [13]. Polynomially solvable special cases of the minimum-concave-cost flow problem include a single-source problem with a single nonlinear arc cost [16], the problem with a

constant number of sources and nonlinear arc costs [23], a production-transportation network flow problem with concave costs defined on a constant number of variables [24], the pure remanufacturing problem [25], and many variants and extensions of the ULSP, which we elaborate in the next paragraph. We also point out here two relatively general classes of polynomially solvable minimum-concave-cost flow problem. Erickson et al. [10] derived a DP algorithm that runs in polynomial time in a planar graph with sources and sinks lying on a constant number of faces of the graph; thus their algorithm runs in polynomial time for the U-MFG with sources in the first row and sinks in the last row, but runs in exponential time for the U-MFG with sources or sinks in more than one rows and the C-MFG in general. Recently, He et al. [17] gave a polynomial-time DP for the U-MFG with a constant number of rows, sources in one row, and sinks in two other rows, and the U-MFG with one source and sinks in a constant number of rows.

The literature on the lot sizing problem is abundant; see the book by Pochet and Wolsey [21]. To be concise, we focus our review on problems that can be formulated as an MFG. We also adopt the terminology in the lot sizing literature: *a time period* refers to a column, and *an echelon* or *a stage* refers to a row in the MFG. For the uncapacitated case, the ULSP was first solved in $O(T^2)$ time by Wagner and Whitin [29], and the complexity was later improved to $O(T \ln T)$ [1, 11, 28]; Zangwill gave an $O(LT^4)$ -time DP for the multi-echelon ULSP with sinks at the last echelon [30]; Love [20] gave an $O(LT^3)$ -time DP algorithm for the multi-echelon ULSP if the production costs are non-increasing over time periods, and the storage costs are non-decreasing over echelons; Zhang et al. [31] recently proposed an $O(T^4)$ -time DP for the two-echelon ULSP with two rows of sinks and fixed setup costs for production, and derived a new class of valid inequalities for the multi-echelon ULSP. For lot sizing problems with production or storage capacity, the capacitated lot sizing problem is NP-hard [7], but the constant capacitated lot sizing problem can be solved in $O(T^3)$ time [12, 26]; the lot sizing problem with variable storage bounds and fixed-charge storage costs can be solved in $O(T^2)$ time [3, 4].

Some extensions of the ULSP are presented in a more general context of supply chain optimization. Kaminsky and Simchi-Levi [19] studied a two-stage ($L = 3$ in our model) supply chain management problem with constant capacity at the first stage, and gave polynomial-time algorithms when the production and transportation costs have several different structures. Van Hoesel et al. [27] extended this two-stage model to a multi-echelon serial supply chain model with general concave production and storage costs, with constant capacity at the first echelon and unlimited capacity at other echelons; they gave an $O(LT^{2L+3})$ -time DP, and the complexity can be significantly reduced if the production and storage costs follow some special patterns. Their result was recently improved by Hwang et al. [18], who developed an $O(LT^8)$ -time algorithm by exploiting a structure called basis path in the optimal solutions.

To the best of our knowledge, all the polynomially solvable capacitated lot sizing and serial supply chain models in the literature make the following two assumptions: (1) the finite capacity is restricted to the production echelon, and there is unlimited capacity at other echelons; (2) the capacity is uniform at all time periods. In our model, this means each vertical arc between the first and second rows of the grid has the same finite capacity, while other arcs have infinite capacity. These assumptions are rather restrictive from a practical point of view. Capacities arise naturally at other echelons of the supply chain, such as transportation between the manufacturer and distribution centers or between distribution centers and retailers, and capacities may vary across time periods as well. It was also pointed out in [19, 27] that production planning models with more general capacity structure is a future research direction and their complexities were posed as open questions. Our result indicates that as long as the number of different capacities over all arcs is a constant, the problem can be solved in polynomial time, regardless of at which stages or upon which vertical or horizontal arcs the capacity is imposed.

The rest of the paper is organized as follows. In Section 2, we reformulate the MFG as an optimal control problem over a dynamical system, propose a general algorithm to solve the problem, and highlight the key component that affects the complexity of the algorithm. In Section 3, we identify two conditions for the C-MFG to be polynomially solvable, and then present several NP-hard cases with any of the conditions relaxed. In Section 4, we identify a general condition for the U-MFG to be polynomially solvable, and complement it with two NP-hard cases. Section 5 discusses extensions of the MFG to grids with additional arcs. Conclusion and some open problem are given in Section 6.

2 Methodology

We introduce some notations and terminology first. Given an integer N , let $[N]$ denote the set $\{1, \dots, N\}$. Let the set P_F denote the *flow polyhedron* defined by constraints in (1). Let $b_{l,t}$ denote the supply of vertex $v_{l,t} \in V$. We call any arc $(v_{l,t}, v_{l,t+1})$ with $l \in [L]$ and $t \in [T-1]$ a *forward arc*, and any arc $(v_{l,t}, v_{l+1,t})$ with $l \in [L-1]$ and $t \in [T]$ a *downward arc*. Given a tree $\mathbb{T} = (V_{\mathbb{T}}, A_{\mathbb{T}})$ and a vertex v , for simplicity we sometimes write $v \in \mathbb{T}$ instead of $v \in V_{\mathbb{T}}$ to denote that v is a vertex in \mathbb{T} . We use G to denote both the grid as well as its planar embedding in Figure 1. The outer face of G refers to the unbounded region outside the rectangle with vertices $v_{1,1}, v_{1,T}, v_{L,1}$, and $v_{L,T}$ in Figure 1.

2.1 Problem reformulation and the algorithm

We always assume the given MFG instance is feasible, since its feasibility can be checked in the same way as in a min-linear-cost flow problem, by solving a maximum flow problem [2]. In order to describe our algorithm, we first reformulate the MFG as an optimal control problem for a discrete-time linear dynamical system with concave costs. The elements of the dynamical system are as follows.

1. Decision stages. There are $T+1$ stages, corresponding to column $t = 1, \dots, T$ in the grid and a dummy stage 0 in the beginning.
2. States. The state s^t at stage $t \in [T-1]$ is an L -dimensional vector whose component s_l^t denotes the flow over the forward arc $(v_{l,t}, v_{l,t+1})$, for $l \in [L]$. Each of the states s^0 and s^T is an L -dimensional zero vector. In addition, the dimension of s^t for $t \in [T-1]$ can be reduced by one, since $\sum_{l \in [L]} s_l^t = \sum_{i \in [t]} \sum_{l \in [L]} b_{l,i}$ according to the flow balance constraints.
3. Decision variables (actions, controls, or inputs). The decision variable u^t at stage $t \in [T]$ is an $(L-1)$ -dimensional vector whose component u_l^t is the flow over the downward arc $(v_{l,t+1}, v_{l+1,t+1})$, for $l \in [L-1]$.
4. The system equations. The state s^{t+1} at stage $t+1$ can be easily determined by s^t and u^t through the flow balance constraints for vertices $v_{1,t+1}, \dots, v_{L,t+1}$. For simplicity, we assume the system equation at stage t is $s^{t+1} = H_t(s^t, u^t)$, where H_t is the affine function representing the flow balance constraints for vertices in column $t+1$.
5. The cost. The cost at stage $t \in [T]$ is the sum of costs incurred by the downward and forward arcs related to that stage. In particular, the cost is $\sum_{l \in [L]} c_{l,t}^f(s_l^t) + \sum_{l \in [L-1]} c_{l,t}^d(u_l^t)$, where $c_{l,t}^f$ is the cost function over forward arc $(v_{l,t}, v_{l,t+1})$, and $c_{l,t}^d$ is the cost function over downward arc $(v_{l,t+1}, v_{l+1,t+1})$.

The above dynamical system is different from the system in [17], where the stages are the anti-diagonal lines of the grid and each component of the state represents the inflow of some vertex. The current dynamical system has the advantage of being easily extended to a grid with horizontal backward arcs and vertical upward arcs. Basically, we only need to augment the state vector and decision variables and change the system equations and costs accordingly. We will discuss several extensions of the MFG in detail in Section 5

The challenge of designing the optimal control over such a dynamical system comes from the uncountable state space and the concave cost structure. The first simplification we can do is to reduce the state space of the system to a finite set. Then the optimal control problem is equivalent to a shortest path problem over an acyclic graph [5]. The simplification is based on the following observation: since the MFG involves minimizing a concave function over the flow polyhedron P_F , so the optimum must be attained at one of its extreme points [22]. Thus it suffices to consider only the states corresponding to those extreme points, and the number of extreme points for any given polyhedron is always finite. We now describe below the algorithm for the MFG.

Algorithm 1 The algorithm for the MFG

1. Enumerate all the possible values of s^t corresponding to the extreme points of P_F , for each $t \in [T - 1]$.
2. Construct a $(T + 1)$ -partite directed graph $G' = (V', A')$. The vertex set $V' = \cup_{t=0}^T V'_t$, where each vertex in V'_t represents a possible value of state s^t (the set V'_0 and V'_T contain exactly one vertex, respectively, since both s^0 and s^T equal the zero vector); each arc in A' goes from one vertex in V'_t to one vertex in V'_{t+1} for $t = 0, \dots, T - 1$. Suppose vertex i in V'_t represents a state $s^{t,i}$ at stage t , vertex j in V'_{t+1} represents a state $s^{t+1,j}$ at stage $t + 1$, and $u^{t,ij} \in \mathbb{R}_+^{L-1}$ is the corresponding decision variable computed through the system equations $s^{t+1,j} = H_t(s^{t,i}, u^{t,ij})$, then the cost of arc (i, j) in G requires $2L - 1$ oracle queries and is calculated as follows:

$$c_{ij} = \sum_{l \in [L]} c_{l,t}^f(s_l^{t,i}) + \sum_{l \in [L-1]} c_{l,t}^d(u_l^{t,ij}),$$

if each component of $u^{t,ij}$ satisfies the corresponding arc capacity constraint, and $c_{ij} = \infty$ otherwise.

3. Find a shortest path P from the vertex representing s^0 to the vertex representing s^T in G' .
 4. Recover a flow of the MFG from the shortest path P , by solving the system equations $s^{t+1} = H_t(s^t, u^t)$ with s^t being the value corresponding to the vertex in V'_t in P , for $t = 0, \dots, T - 1$.
-

Theorem 2.1. *Algorithm 1 solves the MFG in $O(LTM^{2L-2})$ time, where M is the maximum number of flow values a forward arc can attain in all extreme points of P_F .*

Proof. Any extreme point of P_F corresponds to a feasible path from the vertex representing s^0 to the vertex representing s^T in G' , whose length equals the cost of that extreme point; conversely, the shortest path P found by Algorithm 1 corresponds to a feasible flow in the MFG whose cost is equal to the length of the shortest path. Thus the flow recovered from P is an optimal flow for the MFG.

Next we analyze the running time of Algorithm 1. Suppose M is the maximum number of flow values a forward arc can attain in all extreme points of P_F . Since state s^t is an $(L - 1)$ -dimensional

vector with each component being the flow over some forward arc, the number of states at stage t in all extreme points of P_F is $O(M^{L-1})$, and these states can be enumerated in $O(M^{L-1})$ time as well. Then the graph G' has $O(TM^{L-1})$ vertices, and the number of arcs is

$$O(M^{L-1}) + \underbrace{O(M^{2L-2}) + \dots + O(M^{2L-2})}_{T-2 \text{ terms}} + O(M^{L-1}) = O(TM^{2L-2}),$$

where each term in the summation above is the number of arcs between V'_{t-1} and V'_t for $t \in [T]$. The cost of each arc can be evaluated in $O(L)$ time with $O(L)$ oracle queries, so the construction of graph G' takes $O(LTM^{2L-2})$ time including $O(LTM^{2L-2})$ oracle queries. Since graph G' is a directed acyclic graph, the shortest path can be found efficiently in $\Theta(|V'| + |A'|) = O(TM^{2L-2})$ time [9]. It takes $O(LT)$ time to recover of the flow from the found shortest path. The overall running of Algorithm 1 is $O(LTM^{2L-2})$. \square

We should mention that although each component of the state corresponds to a flow value in some extreme point of P_F , the optimal flow output by Algorithm 1 is not necessarily an extreme point, since it is possible for the free arcs in an optimal flow to contain an undirected cycle, unless the optimum is unique.

The remaining task, the key challenge for the MFG, is to find out all possible flow values over a forward arc in all extreme points of P_F , and derive a bound on the value M .

2.2 Characterization of the extreme point of P_F

We first introduce some terminology related to flow and extreme points of P_F , adopted from [2]. Given a feasible flow f , we call an arc a a *free arc* if $0 < f_a < U_a$ and a *restricted arc* if $f_a = 0$ or $f_a = U_a$.

Definition 2.2. [2] A feasible flow f in graph G is a *cycle free solution*, if G contains no undirected cycle composed only of free arcs.

Definition 2.3. [2] A feasible flow f and an associated spanning tree in G is a *spanning tree solution*, if every nontree arc in G is a restricted arc for f .

Note that in a spanning tree solution, the tree arcs can be free or restricted. A tight connection exists between the extreme point of P_F and a cycle free solution (spanning tree solution).

Proposition 2.4 (Chapter 7.3 in [6]). *A feasible flow is an extreme point of P_F if and only if it is a cycle free solution.*

Proposition 2.5. *Given an extreme point of P_F , we can construct a spanning tree solution associated with this extreme point, and the resulting spanning tree solution may not be unique.*

Proof. Consider an extreme f in graph $G = (V, A)$. From Proposition 2.4, f is a cycle free solution. Let A_f be the set of free arcs, then (V, A_f) is a forest. If the forest is a spanning tree, then f and this spanning tree forms a spanning tree solution. Otherwise, we add restricted arcs not in A_f into the forest one at a time, in a way that no undirected cycle is created. In the end, a spanning tree will be produced, and f with the produced spanning tree gives a spanning tree solution. When (V, A_f) is not a spanning tree, the produced spanning tree may contain different restricted arcs, so the spanning tree solution is not unique. \square

We now give a formula to compute the flow over any arc in an extreme point f . We first create a spanning tree solution based on Proposition 2.5. Call the associated spanning tree \mathbb{T}_f . Consider an arbitrary arc $a = (u, v)$ in G . If a is not in \mathbb{T}_f , then the flow over a is zero or U_a . If a is in \mathbb{T}_f , removing a from \mathbb{T}_f breaks \mathbb{T}_f into two connected components. Call the two components sub-trees $\mathbb{T}_{f,1}$ and $\mathbb{T}_{f,2}$, respectively. Without loss of generality, assume that one endpoint u is in $\mathbb{T}_{f,1}$ and another endpoint v is in $\mathbb{T}_{f,2}$. By the flow balance constraints, we have

$$f_a = \sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t} + \sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e, \quad (2)$$

where A_1 is the set of nontree arcs with tails in $\mathbb{T}_{f,1}$ and heads in $\mathbb{T}_{f,2}$, and A_2 is the set of nontree arcs with tails in $\mathbb{T}_{f,2}$ and heads in $\mathbb{T}_{f,1}$.

In the rest of the paper, our main focus is to use (2) to enumerate values of f_a in all extreme points f of P_F . We will present the analysis for the C-MFG first, and then for the U-MFG. We introduce a crucial lemma that will be used in both cases. Below a “path” could refer to either an undirected path or a directed path, when the context is clear. We say two paths are *vertex-disjoint* if the two paths have no vertex in common. The lemma relies on the planar embedding of the grid G .

Lemma 2.6. *Let v^1, v^2, v^3 and v^4 be four distinct vertices lying clockwise on the boundary of the outer face of a plane graph. Suppose P_1 is a path between v^1 and v^3 and P_2 is a path between v^2 and v^4 . Then P_1 and P_2 cannot be vertex-disjoint.*

3 The C-MFG

We first present two set of conditions for the C-MFG to be polynomially solvable, and then complement that with several NP-hard cases when any of the condition is removed.

3.1 Polynomially solvable cases

Theorem 3.1. *The C-MFG can be solved in $O(L^{4KL-4K+1}T^{4KL+4L-4K-3})$ time, if sources and sinks are in at most two rows.*

Proof. First we assume sources and sinks are in row one and row L , i.e., all sources and sinks are on the boundary of the outer face of G . This is without loss of generality. If the sources and sinks are in row l_1 and row l_2 , since all arcs in G are forward and downward arcs, there will be zero flow in each of the arcs above row l_1 and below row l_2 . We can eliminate these arcs from the grid and solve the problem over a smaller grid.

Given an arc a , an extreme point f , and its associated spanning tree,

$$f_a = \sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t} + \sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e,$$

by equation (2). Our goal is to investigate the values of $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ and $\sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e$ in all extremes points, respectively.

Consider the term $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ first. By our assumption, all sources and sinks are on the boundary of the outer face of G . We claim that all sources and sinks in $\mathbb{T}_{f,1}$ appear consecutively. In other words, there exist two vertices v' and v'' in $\mathbb{T}_{f,1}$ such that, if we walk from v' to v'' along the boundary of the outer face of G , then all sources and sinks in $\mathbb{T}_{f,1}$ will be visited with no

source or sink in $\mathbb{T}_{f,2}$ on the way. Thus the possible values for $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ in all extreme points is contained in the set

$$S = \left\{ \sum_{t=i}^j b_{1,t}, \sum_{t=i}^j b_{L,t}, \sum_{t=1}^i b_{1,t} + \sum_{t=1}^j b_{L,t}, \sum_{t=i}^T b_{1,t} + \sum_{t=j}^T b_{L,t}, \text{ for } i, j \in [T] \right\}.$$

The set S can be constructed in $O(T^2)$ time, with cardinality $O(T^2)$. To prove the claim, suppose there exist four vertices v^i ($i = 1, \dots, 4$) lying clockwise on the boundary of the outer face of G , with $v^1, v^3 \in \mathbb{T}_{f,1}$ and $v^2, v^4 \in \mathbb{T}_{f,2}$. Since $\mathbb{T}_{f,1}$ is connected, there is a path between v^1 and v^3 . Similarly there is a path between v^2 and v^4 in $\mathbb{T}_{f,2}$. By Lemma 2.6, the two paths are not vertex-disjoint. But that implies \mathbb{T}_1 and \mathbb{T}_2 are connected, a contradiction.

For the term $\sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e$, each arc e in $A_1 \cup A_2$ is a nontree arc, so $f_e = 0$ or $f_e = U_e$. Suppose there are K different arc capacity values in G . With $O(LT)$ arcs in G , the maximum number of different values for $\sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e$ is $O((LT)^{2K}) = O((LT)^{2K})$, and these values can be enumerated in $O((LT)^{2K})$ time as well.

Therefore, the number of different values for f_a under all extreme points is $O(T^2) \cdot O((LT)^{2K}) = O(L^{2K} T^{2K+2})$. From Theorem 2.1, the C-MFG can be solved in $O(L^{4KL-4K+1} T^{4KL+4L-4K-3})$ time. \square

Corollary 3.2. *The C-MFG with constant L and K and sources and sinks in at most two rows is polynomially solvable.*

If the grid has only two rows (the case for the basic capacitated lot sizing model, studied in [12, 26]), the condition in Corollary 3.2 can be relaxed. Instead of requiring the number of different capacity values over *all* arcs to be constant, we allow the forward arcs to have arbitrary capacities, and require the number of different capacity values over *all downward arcs* to be constant. This condition cannot be further relaxed, since we know the lot sizing problem with general production capacity is NP-hard. Define

$$K_1 = \{U_a \mid U_a < +\infty, a = (v_{1,t}, v_{2,t}) \text{ for some } t \in [T]\}$$

to be the number of different capacity values over downward arcs.

Theorem 3.3. *The C-MFG with two rows and K_1 different capacity values on downward arcs can be solved in $O(T^{4K_1+7})$ time; the problem is polynomially solvable with constant K_1 .*

Proof. The idea of proof is similar to that of Theorem 3.1. Given a forward arc $a = (u, v)$, an extreme point f , and its associated spanning tree,

$$f_a = \sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t} + \sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e,$$

by equation (2). By a similar argument as in Theorem 3.1, the term $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ can take $O(T^2)$ different values. Consider the term $\sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e$. Since the grid only has two rows, the set $A_1 \cup A_2$ can contain at most one forward nontree arc between $\mathbb{T}_{f,1}$ and $\mathbb{T}_{f,2}$ after arc a is deleted, and the flow over that forward arc can take $O(T)$ different values; the set $A_1 \cup A_2$ contains at most T downward arcs, each of which can take K_1 different values. Thus $\sum_{e \in A_2} f_e - \sum_{e \in A_1} f_e$ can take $O(T^{2K_1+1})$ values. Then f_a can take $O(T^{2K_1+3})$ values under all extreme points of P_F . From Theorem 2.1, the C-MFG with $L = 2$ and K_1 different capacity values on downward arcs can be solved in $O(T^{4K_1+7})$ time. \square

3.2 NP-hard cases

In this section, we will show that the problems in Corollary 3.2 and Theorem 3.3 are essentially the most general polynomially solvable C-MFG cases unless $P=NP$. Before stating any results, we want to point out that all proofs for the NP-hard cases in rest of the paper will be accompanied by a figure, to better illustrate the reduction from an instance of an NP-hard problem to an MFG instance. Legends in these figures describe the parameters of the MFG instance: the amount of supply or demand is marked next to the corresponding vertex; the pair (c, U) next to an arc a denotes that the cost of sending any nonzero flow over arc a is c and 0 otherwise, and the capacity of arc a is U ; an arc without such parameters next to it has infinity capacity and zero cost of sending any flow; an arc not present in the figure denotes that it will never be used in any optimal solution because of its large cost. See Figure 2 below for an example.

The next proposition shows the C-MFG becomes NP-hard if sources and sinks are in three rows.

Proposition 3.4. *The C-MFG with $L = 3$, $K = 1$, a single source, and sinks in two other rows is NP-hard.*

Proof. Our proof is based on a reduction from the knapsack problem [14]. The knapsack problem asks that given a set of n items with item i having value y_i and cost c_i for $i = 1, \dots, n$, if there exists a subset of items with cost at most C and total value at least Y . We construct a C-MFG instance with $L = 3$, $K = 1$, a single source, and sinks in two rows as follows. First choose a value $B \geq \max\{y_i \mid i \in [n]\}$. Construct a C-MFG instance with three rows, n columns, one source ($v_{1,1}$ with supply $\sum_{i=1}^n (B - y_i) + Y$), and $n + 1$ sinks ($v_{2,i}$ with demand $(B - y_i)$ for $i \in [n]$ and $v_{3,n}$ with demand Y), as shown in Figure 2. The cost over each arc $(v_{2,i}, v_{3,i})$ has a fixed cost c_i for nonzero flow and 0 otherwise, the cost over each of the rest of arcs in Figure 2 is always 0, and the cost over each of the arcs not present in Figure 2 is always large enough, say $\sum_{i=1}^n c_i + C$, so that they are never used in any optimal solution. The capacity of each downward arc is B , and the capacity of each forward arc is ∞ .

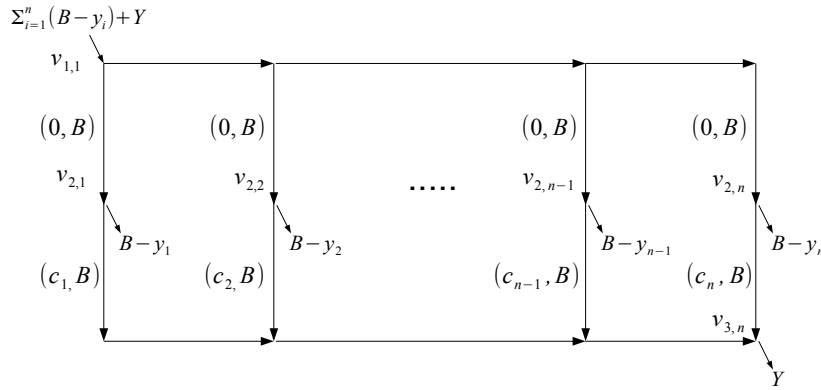


Figure 2: The C-MFG with $L = 3$, $K = 1$, a single source, and sinks in two rows

We claim that the knapsack instance is a yes instance if and only if the optimal objective value of the constructed C-MFG instance is no greater than C . We first show that given any feasible solution of the knapsack instance, we can construct a feasible flow in the C-MFG instance with objective at most C . Let $\epsilon = \sum_{i=1}^n y_i - Y \geq 0$. Since $\epsilon \leq \sum_{i=1}^n y_i$, we can find $\epsilon_1, \dots, \epsilon_n$ such that $0 \leq \epsilon_i < y_i$ for $i \in [n]$ and $\sum_{i=1}^n \epsilon_i = \epsilon$. Given a feasible solution $S \subseteq [n]$ of the knapsack instance with cost $\sum_{i \in S} c_i$, construct a feasible flow in the C-MFG instance as follows: the flow sent over

arc $(v_{1,i}, v_{2,i})$ for $i \notin S$ equals the demand $(B - y_i)$ at $v_{2,i}$; the flow sent over arc $(v_{1,i}, v_{2,i})$ for $i \in S$ equals $B - \epsilon_i$; the flow sent over arc $(v_{2,i}, v_{3,i})$ for $i \in S$ equals $y_i - \epsilon_i$; the flow sent over other arcs are calculated according to the flow conservation constraints. The cost of the constructed feasible flow is $\sum_{i \in S} c_i$, which is at most C . On the other hand, if the optimal objective value of the C-MFG instance is at most C , then the knapsack instance must be a yes instance. Suppose an optimal flow f^* in the C-MFG instance has cost no greater than C . Let f_i^* be the flow over arc $(v_{2,i}, v_{3,i})$ for $i \in [n]$. Let $S = \{i \mid f_i^* > 0\}$. We claim that S is a feasible solution of the knapsack instance, i.e., $\sum_{i \in S} c_i \leq C$ and $\sum_{i \in S} y_i \geq Y$. The cost of flow f^* is $\sum_{i \in S} c_i$, which is no greater than C . Since the capacity of downward arc $(v_{1,i}, v_{2,i})$ is B and the demand at vertex $v_{2,i}$ is $B - y_i$, the maximum amount of flow that can be sent along downward arc $(v_{2,i}, v_{3,i})$ is at most $B - (B - y_i) = y_i$ for $i \in S$. Thus $\sum_{i \in S} y_i \geq \sum_{i \in S} f_i^* = Y$. \square

The capacitated lot sizing problem indicates that the C-MFG with arbitrary capacities on downward arcs is NP-hard. The following proposition shows that the C-MFG with arbitrary capacities on forward arcs is also NP-hard. Both imply the condition that the number of different capacity values K being constant is crucial for the C-MFG to be polynomially solvable.

Proposition 3.5. *The C-MFG with $L = 3$, a single source, a single sink, and arbitrary capacities on forward arcs is NP-hard.*

Proof. Our proof is based on a reduction from the knapsack problem. We construct a C-MFG instance with three rows, a single source $v_{1,1}$ with supply Y , and a single sink $v_{3,2n}$ with demand Y , as shown in Figure 3. The forward arc $(v_{2,2i-1}, v_{2,2i})$ has a capacity y_i , and a cost of c_i of sending any nonzero flow and 0 otherwise, for $i \in [n]$. Each of the remaining arcs in Figure 3 has zero cost of sending any flow and an infinite capacity.

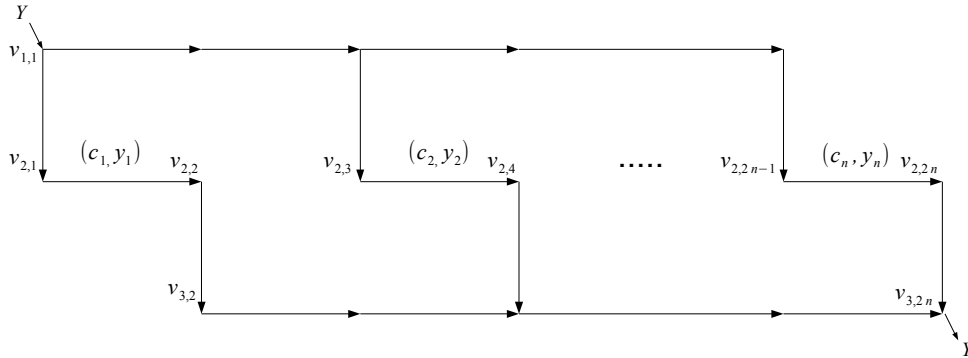


Figure 3: The C-MFG with three rows, a single source, a single sink, and general capacity on forward arcs

We claim that the knapsack instance is a yes instance if and only if the optimal objective value of the constructed C-MFG instance is no greater than C . Let $S \subseteq [n]$ be a feasible solution of the knapsack instance, i.e., $\sum_{i \in S} c_i \leq C$ and $\sum_{i \in S} y_i \geq Y$. Let $\epsilon = \sum_{i=1}^n y_i - Y \geq 0$. Since $\epsilon \leq \sum_{i=1}^n y_i$, we can find $\epsilon_1, \dots, \epsilon_n$ such that $0 \leq \epsilon_i < y_i$ for $i = 1, \dots, n$ and $\sum_{i=1}^n \epsilon_i = \epsilon$. Construct a feasible flow in the C-MFG instance as follows: the flow sent over arc $(v_{2,2i-1}, v_{2,2i})$ is $y_i - \epsilon_i$ for $i \in S$; the flow sent over other arcs are calculated according to the flow conservation constraints. The cost of the constructed feasible flow is $\sum_{i \in S} c_i$, which is at most C . On the other hand, let f^* be an optimal flow of the C-MFG instance and f_i^* be the flow over arc $(v_{2,2i-1}, v_{2,2i})$ for $i \in [n]$. Let $S = \{i \mid f_i^* > 0\}$. Then it is easy to verify that S is a feasible solution of the knapsack instance. \square

Remark 3.6. The conditions for a capacitated lot sizing model to be polynomially solvable are very subtle. By Proposition 3.4, the *constant capacitated* multi-echelon lot sizing problem with intermediate demands is NP-hard, in contrast to the polynomially solvable *uncapacitated* multi-echelon lot sizing problem with intermediate demands [31, 17]. By Proposition 3.5, the uncapacitated *multi-echelon* lot sizing problem with variable storage bounds is NP-hard, in contrast to the polynomially solvable *single-echelon* lot sizing problem with variable storage bounds [4].

4 The U-MFG

We first present a general condition for the U-MFG to be polynomially solvable, and then complement that with several NP-hard cases.

4.1 Polynomially solvable cases

Theorem 4.1. *The U-MFG with sources in one row can be solved in $O(LT^{8L^2-12L+5})$ time.*

Theorem 4.1 immediately implies the following result.

Corollary 4.2. *The U-MFG with sources in one row and a constant number of rows is polynomially solvable.*

The main ingredient to prove Theorem 4.1 is to enumerate all possible flow values over any arc in all extreme points of P_F . This is summarized in the following proposition.

Proposition 4.3. *In the U-MFG with sources in one row, the flow over any arc can take $O(T^{4L-2})$ different values in all extreme points of P_F , and these values can be enumerated in $O(T^{4L-2})$ time.*

Theorem 4.1 then follows immediately from Theorem 2.1 and Proposition 4.3. The proof of Proposition 4.3 is slightly technical and requires several new gadgets, which we introduce below. The proof can be summarized in two steps: (1) For each extreme point of P_F , construct a corresponding spanning tree solution with certain property; (2) With the constructed spanning tree solution, the flow over any arc can be written as a sum of supplies at vertices following a special pattern. Then we can enumerate the possible flow values in all extreme points in an efficient way, instead of simply enumerating all extreme points and calculating the flow values accordingly.

4.1.1 A special spanning tree solution

We first assume that all sources are in row one. Otherwise, we can remove all arcs in rows above the sources, without changing the optimal flow over the remaining arcs. We need the following concept before constructing the spanning tree with the required property.

Definition 4.4. Given a tree of the grid, a vertex is said to be *accessible from row one*, if there exists a directed path in the tree starting from some vertex in row one to that vertex.

Consider the spanning tree in Figure 4, the vertex v_{l,i_1} is accessible from row one, since there is a directed path from a first-row vertex v_{1,i_1} to v_{l,i_1} . In fact, each vertex in that tree is accessible from row one. We will construct a spanning tree with this property for every extreme point of P_F . In particular, given an extreme point f , let $A_f = \{e \mid f_e > 0\}$. By Proposition 2.4, (V, A_f) is a forest. Our goal is to (possibly) include some restricted arcs not in A_f to produce a spanning tree T_f with the following property.

Algorithm 2 Construct a spanning tree solution satisfying Property (A1)

Input: An extreme point f of P_F

Output: A spanning tree $\mathbb{T}_f = (V, A_{\mathbb{T}_f})$ satisfying Property (A1)

Initialization: $A_{\mathbb{T}_f} \leftarrow A_f$

while $(V, A_{\mathbb{T}_f})$ is not a spanning tree **do**

Select an arc $e = (u_e, v_e) \notin A_{\mathbb{T}_f}$ such that u_e is accessible from row one and $(V, A_{\mathbb{T}_f} \cup \{e\})$ does not contain an undirected cycle.

$A_{\mathbb{T}_f} \leftarrow A_{\mathbb{T}_f} \cup \{e\}$

end while

(A1) Every vertex v in G is accessible from row one.

This can be achieved by Algorithm 2.

Proposition 4.5. *The spanning tree constructed by Algorithm 2 satisfies Property (A1).*

Proof. During the construction of the tree, we call a vertex in G *isolated* if the vertex is not adjacent to any arc in $A_{\mathbb{T}_f}$ and *non-isolated* otherwise. We prove the result by induction on the number of non-isolated vertices. During the initialization step, all non-isolated vertices, i.e., vertices incident to at least one arc in A_f , are accessible from row one, due to the flow conservation constraints and the fact that all sources are in row one. At each iteration, if there exists an arc $e = (u_e, v_e)$ not in $A_{\mathbb{T}_f}$ such that u_e is non-isolated and v_e is isolated, then add arc e into $A_{\mathbb{T}_f}$. Since u_e is accessible from row one, vertex v_e becomes non-isolated and is accessible from row one as well. If such an arc does not exist, then there must exist an isolated vertex v that is in row one. Select one of the two outgoing arcs of vertex v as arc e , and include it into $A_{\mathbb{T}_f}$. Then v is accessible from row one by definition. \square

The constructed spanning tree has some additional nice properties. To establish those properties, we first define an auxiliary function $\kappa_l : [T] \rightarrow [T]$ for each $l \in [L]$.

Definition 4.6. Given a spanning tree \mathbb{T}_f with Property (A1), for each $l \in [L]$,

1. the value $\kappa_l(i)$ is the smallest integer j such that there is a directed path from $v_{1,j}$ to $v_{l,i}$ in \mathbb{T}_f .
2. the path $P_{l,i}$ denotes the unique directed path in \mathbb{T}_f from $v_{1,\kappa_l(i)}$ to $v_{l,i}$.

For the example in Figure 4, $\kappa_l(i_1) = i_1$, $\kappa_l(i_2) = \dots = \kappa_l(i_6) = i_2$, and $\kappa_l(i_7) = i_3$. The path P_{l,i_1} consists of downward arcs $(v_{1,i_1}, v_{2,i_1}), \dots, (v_{l-1,i_1}, v_{l,i_1})$, and the path P_{l,i_7} consists of forward arcs $(v_{1,i_3}, v_{1,i_4}), \dots, (v_{1,i_6}, v_{1,i_7})$ and downward arcs $(v_{1,i_7}, v_{2,i_7}), \dots, (v_{l-1,i_7}, v_{l,i_7})$.

Lemma 4.7.

1. For each $l \in [L]$, the function κ_l is non-decreasing.
2. The union of paths $\{P_{l,i} : i \in [T]\}$ forms a forest of arborescences.

Proof. Fix $l \in [L]$. Suppose for some $i < i'$ we have $\kappa_l(i) = j > \kappa_l(i') = j'$. Consider the two directed paths $P_{l,i}$ and $P_{l,i'}$ in \mathbb{T}_f . We claim they cannot be vertex-disjoint. To see this, first the two paths do not contain any arcs below row l , since all arcs are either forward or downward in G . Consider the subgraph of G consisting of only vertices and arcs in row one to row l . It is also planar, and vertices $v_{1,j'}, v_{1,j}, v_{l,i'}$, and $v_{l,i}$ lie clockwise on the boundary of its outer face. By

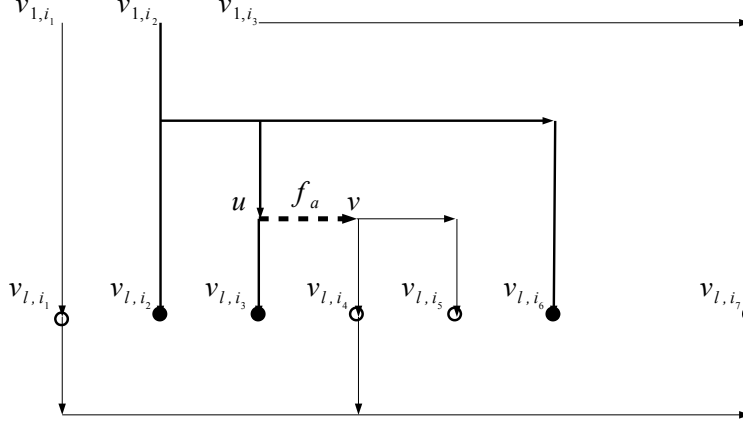


Figure 4: The spanning tree \mathbb{T}_f satisfying Property (A1), the given arc $a = (u, v)$, and vertices of type 1 (hollow dots) and type 2 (solid dots) in row l

Lemma 2.6, the two paths $P_{l,i}$ and $P_{l,i'}$ cannot be vertex-disjoint. Then there will be a directed path from $v_{1,j'}$ to $v_{l,i}$ in \mathbb{T}_f , contradicting the definition of $\kappa_l(i)$.

To prove the second statement, it suffices to show that if $P_{l,i}$ and $P_{l,i'}$ are not vertex-disjoint, then they must start from the same vertex in the first row. Suppose they start from two different vertices, $v_{1,j}$ and $v_{1,j'}$ with $j < j'$. Since the two paths are not vertex-disjoint, there is a directed path from $v_{1,j}$ to $v_{l,i'}$. The path $P_{l,i'}$ should start from $v_{1,j}$ instead of $v_{1,j'}$, a contradiction. \square

Consider the example in Figure 4. For vertices in row l , $\kappa_l(i_1) = i_1$, $\kappa_l(i_2) = \dots = \kappa_l(i_6) = i_2$, and $\kappa_l(i_7) = i_3$, so κ_l is non-decreasing; the union of paths $P_{l,i}$ ($i \in [T]$) forms three arborescences: the directed path from v_{1,i_1} to v_{l,i_1} , the arborescence rooted at v_{1,i_2} with leaves v_{l,i_2} to v_{l,i_6} , and the directed path from v_{1,i_3} to v_{l,i_7} .

4.1.2 The flow value f_a

It is known that if we delete an arc $a = (u, v)$ from a spanning tree \mathbb{T}_f , the tree will break into two subtrees $\mathbb{T}_{f,1}$ (containing vertex u) and $\mathbb{T}_{f,2}$ (containing vertex v). We classify all vertices in V according to their locations in the subtrees.

Definition 4.8. Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1),

1. a vertex is of *type 1* if it is contained in $\mathbb{T}_{f,1}$, and of *type 2* if it is contained in $\mathbb{T}_{f,2}$;
2. a type 2 vertex is of *type 2A*, if the directed path in \mathbb{T}_f from the vertex in row one to that vertex contains arc a ; a type 2 vertex is of *type 2B* otherwise.

Consider the example in Figure 4 with arc $a = (u, v)$. For vertices in row one, vertices v_{1,i_1} and v_{1,i_3} are of type 2, and vertex v_{1,i_2} is of type 1. For vertices in row l , vertices v_{l,i_2} , v_{l,i_3} , and v_{l,i_6} are of type 1, v_{l,i_4} and v_{l,i_5} are of type 2A, and v_{l,i_1} and v_{l,i_7} are of type 2B.

We introduce several lemmas to characterize the location of vertices of each type in a given row. These lemmas will be used later for enumerating the values of f_a in all extreme points.

Lemma 4.9. Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), there cannot be four vertices in row one that alternate between type 1 and type 2.

Proof. Suppose there exist $v_{1,i}, v_{1,j}, v_{1,i'},$ and $v_{1,j'}$ with $i < j < i' < j'$ such that $v_{1,i}$ and $v_{1,i'}$ are of the same type and $v_{1,j}$ and $v_{1,j'}$ are of another type. WLOG, assume $v_{1,i}$ and $v_{1,i'}$ are of type 1 and $v_{1,j}$ and $v_{1,j'}$ are of type 2. Then there is an undirected path between $v_{1,i}$ and $v_{1,i'}$ in $\mathbb{T}_{f,1}$ and an undirected path between $v_{1,j}$ and $v_{1,j'}$ in $\mathbb{T}_{f,2}$. These two paths cannot be vertex-disjoint by Lemma 2.6. That makes $\mathbb{T}_{f,1}$ and $\mathbb{T}_{f,2}$ connected, a contradiction. \square

Lemma 4.10. *Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), if $v_{l,i}$ and $v_{l,j}$ are of type 2A with $i < j$, then $v_{l,k}$ is also of type 2A, for any k such that $i < k < j$ and each $l \in [L]$.*

Proof. Consider the directed paths $P_{l,i}$ and $P_{l,j}$ that go from some vertices in row one to $v_{l,i}$ and $v_{l,j}$ in \mathbb{T}_f , respectively. By the definition of type 2A vertices, both $P_{l,i}$ and $P_{l,j}$ contain arc a . Then $\kappa_l(i) = \kappa_l(j)$, following from the second statement of Lemma 4.7. Since $i < k < j$, $\kappa_l(i) \leq \kappa_l(k) \leq \kappa_l(j)$ by the first statement of Lemma 4.7. Then $\kappa_l(i) = \kappa_l(k) = \kappa_l(j)$. The directed paths that connect vertices in row one to $v_{l,i}$, $v_{l,k}$, and $v_{l,j}$ start from the same vertex $v_{1,\kappa_l(i)}$ in row one. Since the paths $P_{l,i}$ and $P_{l,j}$ both contain arc a and there is only one path in \mathbb{T}_f from vertex $v_{1,\kappa_l(i)}$ to vertex v (the head of a), the segment in $P_{l,i}$ from $v_{1,\kappa_l(i)}$ to v should be the same as the segment in $P_{l,j}$ from $v_{1,\kappa_l(i)}$ to v , which is the same as the segment in $P_{l,k}$ from $v_{1,\kappa_l(i)}$ to v as well. Thus $P_{l,k}$ contains arc a and $v_{l,k}$ is also of type 2A. \square

Lemma 4.11. *Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), there cannot be four vertices in row l ($l > 1$) that alternate between type 1 and type 2B.*

Proof. Suppose there exist $v_{l,i}, v_{l,j}, v_{l,i'},$ and $v_{l,j'}$ with $i < j < i' < j'$ such that $v_{l,i}$ and $v_{l,i'}$ are of the same type and $v_{l,j}$ and $v_{l,j'}$ are of another type. WLOG, assume $v_{l,i}$ and $v_{l,i'}$ are of type 1 and $v_{l,j}$ and $v_{l,j'}$ are of type 2B. Consider the four directed paths $P_{l,i}, P_{l,j}, P_{l,i'}, P_{l,j'}$ in \mathbb{T}_f . None of these paths contains arc a according to the definitions of type 1 and type 2B vertices, so $P_{l,i}$ and $P_{l,i'}$ are contained entirely in $\mathbb{T}_{f,1}$, and $P_{l,j}$ and $P_{l,j'}$ are contained entirely in $\mathbb{T}_{f,2}$. Consider the starting vertices of the four directed paths: $v_{1,\kappa_l(i)}$ and $v_{1,\kappa_l(j)}$ are of type 1, and $v_{1,\kappa_l(i')}$ and $v_{1,\kappa_l(j')}$ are of type 2; moreover, $\kappa_l(i) < \kappa_l(j) < \kappa_l(i') < \kappa_l(j')$ by Lemma 4.7. Thus we found four vertices in row one that alternate between type 1 and type 2, which contradicts Lemma 4.9. \square

Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), the vertices in row l for each $l \in [L]$ can be partitioned into groups of consecutive type 1 and type 2 vertices.

Definition 4.12. Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), an *interval of type 1 (type 2, type 2A, type 2B)* in row l ($l \in [L]$) is a set of adjacent vertices $\{v_{l,i} \mid i_1 \leq i \leq j_1\}$ of type 1 (type 2, type 2A, type 2B). An interval is *maximal* if it is not contained in any larger interval of the same type.

Consider the example in Figure 4. Vertices v_{l,i_2} and v_{l,i_3} form a maximal interval of type 1, and vertices v_{l,i_4} and v_{l,i_5} form a maximal interval of type 2A. The following lemma shows that the number of maximal intervals in each row is small.

Lemma 4.13. *Given an arc $a = (u, v) \in A$ and a spanning tree \mathbb{T}_f with Property (A1), row l has at most two maximal intervals of type 2, or at most two maximal intervals of type 1, for each $l \in [L]$.*

Proof. When $l = 1$, there are at most two maximal intervals of type 1 and two maximal intervals of type 2, according to Lemma 4.9. For $l > 1$, we consider two different cases, depending on whether row l contains any type 2A vertex or not. If row l does not contain any vertex of type

2A, then it can contain at most two maximal intervals of type 2B, according to Lemma 4.11. If row l contains at least one type 2A vertex, then all type 2A vertices form a maximal interval of type 2A, according to Lemma 4.10. Suppose there are at least three maximal intervals of type 1 and three maximal intervals of type 2 in row l . We already know one maximal interval of type 2 contains all type 2A vertices in that row. For the remaining five intervals, we can always find four vertices, $v_{l,i}, v_{l,j}, v_{l,i'},$ and $v_{l,j'}$ with $i < j < i' < j'$, that alternate between type 1 and type 2B, which contradicts Lemma 4.11. \square

Consider the example in Figure 4. Vertices in row l present in Figure 4 are divided into two types: $v_{l,i_2}, v_{l,i_3},$ and v_{l,i_6} are of type 1, and vertices $v_{l,i_1}, v_{l,i_4}, v_{l,i_5},$ and v_{l,i_7} are of type 2. Row l contains only two maximal intervals of type 1.

Now with all the gadgets and lemmas introduced above, we are ready to prove Proposition 4.3.

Proof of Proposition 4.3. Given an arc $a = (u, v) \in A$ and an extreme point f , we first construct a spanning tree \mathbb{T}_f with Property (A1) by Algorithm 2. Then the flow over arc a

$$f_a = \sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t} \quad (3a)$$

$$= \sum_{v_{1,t} \in \mathbb{T}_{f,1}} b_{1,t} + \sum_{l=2}^L \sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}, \quad (3b)$$

by (2). By Lemma 4.9, the first term in (3b), $\sum_{v_{1,t} \in \mathbb{T}_{f,1}} b_{1,t}$, equals $\sum_{i \leq t \leq j} b_{1,t}$ or $\sum_{1 \leq t \leq i} b_{1,t} + \sum_{j \leq t \leq T} b_{1,t}$, for some $i, j \in [T]$ with $i \leq j$. Thus the term $\sum_{v_{1,t} \in \mathbb{T}_{f,1}} b_{1,t}$ takes $O(T^2)$ values in all extreme points of P_F , and these values can be enumerated in $O(T^2)$ time. For the second term in (3b), by Lemma 4.11, $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ equals either $\sum_{i_1 \leq t \leq j_1} b_{l,t} + \sum_{i_2 \leq t \leq j_2} b_{l,t}$ or $\sum_{t \in [T]} b_{l,t} - (\sum_{i_1 \leq t \leq j_1} b_{l,t} + \sum_{i_2 \leq t \leq j_2} b_{l,t})$ for some $i_1, j_1, i_2, j_2 \in [T]$ with $i_1 \leq j_1 \leq i_2 \leq j_2$. Thus the term $\sum_{v_{l,t} \in \mathbb{T}_{f,1}} b_{l,t}$ takes $O(T^4)$ values, and these values can be enumerated in $O(T^4)$ time.

Therefore, the flow f_a takes $O(T^2) \times O(T^{4(L-1)}) = O(T^{4L-2})$ values in all extreme points of P_F , and these values can be enumerated in $O(T^{4L-2})$ time. \square

4.2 NP-hard cases

Each condition in Corollary 4.2 is crucial for the U-MFG to be polynomially solvable. As shown below, the U-MFG becomes NP-hard if any of those conditions is relaxed.

Proposition 4.14. *The U-MFG with sources in one row and a varying number of rows is NP-hard.*

Proof. Our proof is based on a reduction from the partition problem [14]. Given a partition instance with a set of n integers $\{y_1, \dots, y_n\}$, we construct a U-MFG instance with $n+1$ rows, $n+1$ columns, two sources in row one, and n sinks, as shown in Figure 5. The grid has two sources ($v_{1,1}$ and $v_{1,2}$, each with supply $\sum_{i \in [n]} y_i/2$) and n sinks ($v_{i+1,i+1}$ with demand y_i for each $i \in [n]$). The cost over each incoming arc of sink $v_{i+1,i+1}$ ($(v_{i+1,i}, v_{i+1,i+1})$ or $(v_{i,i+1}, v_{i+1,i+1})$) is 1 for sending nonzero flow and 0 otherwise for each $i \in [n]$. Each of the remaining arcs in Figure 5 has zero cost of sending any flow.

We claim that the partition instance is a yes instance if and only if the optimal objective value of the constructed U-MFG instance is n . First notice the optimal objective value of the U-MFG instance is at least n , since at least one incoming arc of sink $v_{i+1,i+1}$ needs to carry nonzero flow, for each $i \in [n]$. If there exists a subset $S \subseteq [n]$ such that $\sum_{i \in S} y_i = \sum_{i \in [n]} y_i/2$, then we can create

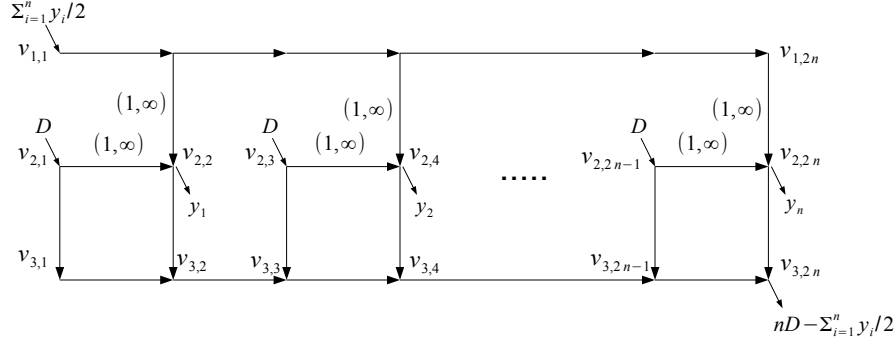


Figure 6: The MFG instance with both sources and sinks in two rows

The reformulation and algorithm in Section 2 can be easily modified for the MFG-BU, but the computational complexity results in Section 3 and Section 4 vary across different cases. We illustrate the differences below.

First the dynamical system reformulated from the MFG-BU will include flow over backward arcs into states and flow over upward arcs into decision variables. In particular, the state s^t at stage t becomes a $2L$ -dimensional vector whose first L components denote the flows over forward arcs $(v_{l,t}, v_{l,t+1})$ and last L components denote the flows over backward arcs $(v_{l,t+1}, v_{l,t})$, for $l \in [L]$; the decision variable u^t at stage $t \in [T]$ becomes a $2(L-1)$ -dimensional vector whose first $(L-1)$ components denote the flows over downward arcs $(v_{l,t+1}, v_{l+1,t+1})$ and last $(L-1)$ components denote the flows over upward arcs $(v_{l+1,t+1}, v_{l,t+1})$, for $l \in [L-1]$. The system equations and cost calculation need to be modified accordingly. The algorithm for the MFG-BU remains almost the same as Algorithm 1, with the only change being to include costs over backward and upward arcs in the calculation of arc costs; its complexity is slightly higher than that in Theorem 2.1 (up to a constant factor in the exponent), and the analysis is similar.

Theorem 3.1 no longer holds for the capacitated MFG-BU. The key reason is that we cannot assume sources and sinks are in row one and row L (or the boundary of the outer face of the grid) any more, due to the existence of upward arcs. Thus Lemma 2.6 cannot be applied, and the sources and sinks in subtree $\mathbb{T}_{f,1}$ do not appear consecutively. Corollary 3.2 still holds though for the capacitated MFG-BU if there is no upward arc in the grid. Similarly, the uncapacitated MFG-BU with no upward arcs, sources in one row and a constant number of rows is still polynomially solvable. It becomes NP-hard when the grid contains upward arcs, as shown below.

Proposition 5.1. *The U-MFG with upward arcs and three rows is NP-hard.*

Proof. Our proof is based on a reduction from the partition problem. Given a partition instance with a set of n integers $\{y_1, \dots, y_n\}$, we construct a U-MFG instance with three rows and $n+1$ column, as shown in Figure 7. The grid has two sources ($v_{1,1}$ with supply $\sum_{i \in [n]} y_i/2$ and $v_{1,2}$ with supply $\sum_{i \in [n]} y_i/2$) and n sinks ($v_{2,i+1}$ with demand y_i for each $i \in [n]$). The cost over each incoming arc of sink $v_{2,i+1}$ ($(v_{1,i+1}, v_{2,i+1})$ or $(v_{3,i+1}, v_{2,i+1})$) is 1 for sending nonzero flow and 0 otherwise, for each $i \in [n]$. Each of the remaining arcs in Figure 7 has zero cost of sending any flow. It can be seen that the partition instance is a yes instance if and only if the optimal objective value of the constructed U-MFG instance is n , following a similar argument as in the proof of Proposition 4.14. \square

We can even extend our polynomially solvable results to grids with diagonal arcs of the form $(v_{l,t}, v_{l+1,t+1})$ or $(v_{l,t+1}, v_{l+1,t})$, as long as adding these arcs keeps the grid planar. Then Lemma 2.6

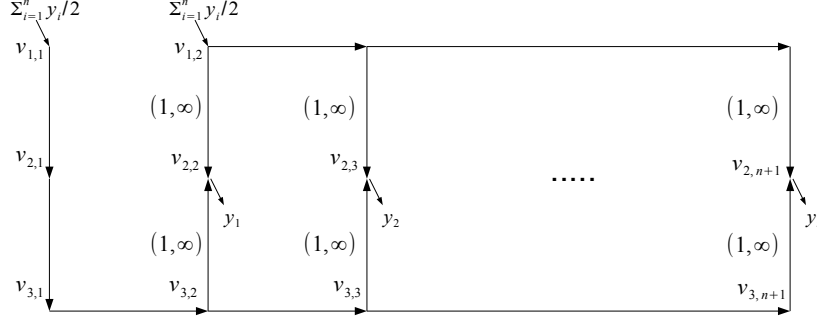


Figure 7: The MFG instance with upward arcs

still holds, and the reformulation and analysis of the algorithm can be done in a similar fashion as the inclusion of backward arcs. Finally, throughout the paper we studied the computational complexity of the MFG when the number of *columns* in the grid, T , varies. By symmetry, if we let the number of rows L varies and put the corresponding conditions on columns (such as all sources are in one column), all results hold accordingly.

6 Conclusions

We establish a full characterization of the computational complexity of the minimum-concave-cost flow problem in a two-dimensional grid, based on the number of rows of the grid, the number of different capacities over all arcs, and the location of sources and sinks. We develop polynomial-time algorithms for the problem under several general conditions, by exploiting the fact of the grid being planar and the combinatorial structure underlying the extreme points of the flow polyhedron. We also complement these results with hardness results when any of the conditions is relaxed. Our results answer several open questions raised in the lot sizing and supply chain literature. An interesting open problem left is the computational complexity of the C-MFG with constant K , varying L , and sources and sinks in at most two rows, which we conjecture to be NP-hard.

References

- [1] A. Aggarwal and J. K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41(3):549–571, 1993.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, Upper Saddle River, 1993.
- [3] A. Atamtürk and S. Küçükyavuz. Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation. *Operations Research*, 53(4):711–730, 2005.
- [4] A. Atamtürk and S. Küçükyavuz. An $O(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs. *Operations Research Letters*, 36(3):297–299, 2008.
- [5] D. P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, Massachusetts, 1996.
- [6] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific Belmont, Massachusetts, 1997.

- [7] G. R. Bitran and H. H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1982.
- [8] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *The Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.
- [9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms, Third Edition*. MIT Press, 2009.
- [10] R. E. Erickson, C. L. Monma, and A. F. Veinott, Jr. Send-and-split method for minimum-concave-cost network flows. *Mathematics of Operations Research*, 12(4):634–664, 1987.
- [11] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time. *Management Science*, 37(8):909–925, 1991.
- [12] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18(1):12–20, 1971.
- [13] D. B. Fontes, E. Hadjiconstantinou, and N. Christofides. A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. *European Journal of Operational Research*, 174(2):1205–1219, 2006.
- [14] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.
- [15] G. M. Guisewite and P. M. Pardalos. Minimum concave-cost network flow problems: applications, complexity, and algorithms. *Annals of Operations Research*, 25(1):75–99, 1990.
- [16] G. M. Guisewite and P. M. Pardalos. A polynomial time solvable concave network flow problem. *Networks*, 23(2):143–147, 1993.
- [17] Q. He, S. Ahmed, and G. L. Nemhauser. Minimum concave cost flow over a grid network. *Mathematical Programming*, 150(1):79–98, 2015.
- [18] H.-C. Hwang, H.-S. Ahn, and P. Kaminsky. Basis paths and a polynomial algorithm for the multistage production-capacitated lot-sizing problem. *Operations Research*, 61(2):469–482, 2013.
- [19] P. Kaminsky and D. Simchi-Levi. Production and distribution lot sizing in a two stage supply chain. *IIE Transactions*, 35(11):1065–1075, 2003.
- [20] S. F. Love. A facilities in series inventory model with nested schedules. *Management Science*, 18(5-part-1):327–338, 1972.
- [21] Y. Pochet and L. A. Wolsey. *Production planning by mixed integer programming*. Springer Verlag, New York, 2006.
- [22] R. T. Rockafellar. *Convex analysis*. Princeton university press, 2015.
- [23] H. Tuy, S. Ghannadan, A. Migdalas, and P. Värbrand. The minimum concave cost network flow problem with fixed numbers of sources and nonlinear arc costs. *Journal of Global Optimization*, 6(2):135–151, 1995.

- [24] H. Tuy, S. Ghannadan, A. Migdalas, and P. Värbrand. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Mathematical Programming*, 72(3):229–258, 1996.
- [25] W. van den Heuvel and A. P. M. Wagelmans. Four equivalent lot-sizing models. *Operations Research Letters*, 36(4):465–470, 2008.
- [26] C. P. M. van Hoesel and A. P. M. Wagelmans. An $O(T^3)$ algorithm for the economic lot-sizing problem with constant capacities. *Management Science*, 42(1):142–150, 1996.
- [27] S. van Hoesel, H. E. Romeijn, D. R. Morales, and A. P. M. Wagelmans. Integrated lot sizing in serial supply chains with production capacities. *Management science*, 51(11):1706–1719, 2005.
- [28] A. P. M. Wagelmans, S. van Hoesel, and A. Kolen. Economic lot sizing: an $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40(1):145–156, 1992.
- [29] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.
- [30] W. I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science*, 15(9):506–527, 1969.
- [31] M. Zhang, S. Küçükyavuz, and H. Yaman. A polyhedral study of multi-echelon lot sizing with intermediate demands. *Operations Research*, 60(4):918–935, 2012.